Predicting U.S. Domestic Air Travel Delays

Patrick Sollars, Aaron Newman, and Cecilia Chen

Introduction

No one likes flight delays. Unfortunately, they continue to happen, and while they are rare, they happen more frequently than we would prefer. The goal of this project is to better understand the factors that contribute to delays in United States domestic air travel and to see whether we can effectively predict the likelihood of a delay in the future.

Successful delay prediction would allow airlines, airports, air traffic control organizations, and passengers to make better choices that lessen the likelihood and/or impact of delay. A passenger may move travel by a day or two if a lesser delay is predicted. An airline may choose to pre-position more cleaning crews at an impacted airport to turn around aircraft more quickly for the next flight. Over time, these choices could lead to long-term optimizations that reduce delays overall.

We chose to pursue this project because we all have personal and/or professional interest in commercial aviation. We all travel by air, we have all experienced travel delays, and we'd like to avoid them in the future. In August 2023, one of our team members endured a 17-hour delay on a flight from New York City to Washington, D.C. That experience certainly drove our desire for a solution.

We employed two distinct feature selector methods, XGBoost and Random Forest importance, from the Python library AutoFeatSelect for feature selection. These algorithms successfully reduced the initial set of over 100 features to just 8 relevant features. Following this feature selection process, we further employed Principal Component Analysis (PCA) to reduce the dimensionality of the data down to 4 principal components.

We validated the selected features across various supervised learning models by employing the LazyClassifier model. This validation process yielded promising results. When we tested these feature selection results on a smaller dataset, specifically, the 2019 data for one airline, the model's performance appeared satisfactory. However, when we applied a different feature selection method to the same smaller dataset, we observed that two of the initially selected features were excluded before the feature selection process. This observation suggests that the selected features may differ among airlines when predicting delay labels, possibly reflecting variations in their operations, target customer profiles, and other factors. In addition, the outcome of feature selection suggests that the features within the FAA dataset may not exert a substantial influence on the prediction of the delay label, which differs from what we initially presumed.

To perform the binary classification, we trained 6 different models on the selected dataset. While the results were lackluster, the best performing model, CatBoost, was selected for a follow up analysis where we tuned the hyper-parameters using grid search with cross validation. The best fit model was only able to score 0.50 on the balanced F1 metric, our primary evaluation target. Again, this suggests that the data that we had initially selected did not provide the predictive power that we had hoped for.

For our time series prediction work, we found that XGBoost gave us the best performance for predicting the number of delayed flights in a given day. (This model also did not rely on features from the FAA dataset.) However, the mean error for the model is high enough that we can't recommend its use in a production environment. There are likely unobserved influences on delays that we were not able to model.

Related Work

One of our team members participated in a Milestone I project using similar data to study the impact of COVID-19 infection rates on air travel passenger volumes. The project was able to use Vector Autoregression (VAR) to make reasonable predictions, but the monthly data used was not sufficiently granular to use more advanced time-series

prediction techniques. This Milestone II project used a more granular dataset to answer different questions. See [1] for more information about the Milestone I research.

We have discovered that there are quite a few papers, articles, and Kaggle projects using similar air travel data. One such Kaggle tutorial was published in 2017 by Daniel Fabien [2]. Fabien's tutorial helped us become more familiar with the structure of the data, but his analysis was limited to just one month of travel. We were disappointed to discover a paper that is still available online that appears to directly use information from the Kaggle tutorial without citation. [3] Another instructive article by Javier Herbas was published as a *Medium* post in 2020. [4] Herbas' work gave us some insight on features that may have limited predictive value.

Our Milestone research is distinct from prior published research because we are using unsupervised machine learning techniques to assist in feature selection and engineering. We are attempting to broaden the use of data to a larger time scale, and not limited to a specific geography. We are also adding additional potential features from a complementary dataset. Finally, we are exercising time-series prediction techniques that are not prevalent in the prior research.

Data Sources & Pre-Processing

Our primary dataset for this effort is Airline On-Time Performance Data managed by the U.S. Department of Transportation's Bureau of Transportation Statistics (BTS). On a monthly basis, participating U.S. airlines report the status of each one of their flights to BTS, which includes the actual times of departure and arrival, the scheduled times of departure and arrival, and a host of additional information about the flight. This information includes the flight number, origin and destination, and the registration number (also called tail number or N-number) of the aircraft. If the flight did experience a delay, information is provided about the cause such as carrier, weather, National Air System (NAS), security, or late inbound aircraft. See [5] for more general information about this dataset.

There is no application programming interface (API) for this dataset, so it was necessary to download zip archives for each month from January 2015 through June 2023 (the most recent data available at this writing). Each zip archive contains a comma-separated values (CSV) file for that month with 109 columns. More information about the column contents is provided in the appendix. We downloaded 102 monthly files for a total of 2.66 gigabytes, which increased to 23.66 gigabytes when uncompressed. There are 52,493,035 total records in the dataset.

We initially read the data into a sqlite database table. The resulting database was almost 18 gigabytes in size but was later reduced to 12 gigabytes by quickly removing columns that were sparse and determined to not represent influential features, and making sure that column types were set properly. As we progressed with our analysis, we found it more effective to interact with the data in parquet files, which were between 1.5 and 3 gigabytes in size.

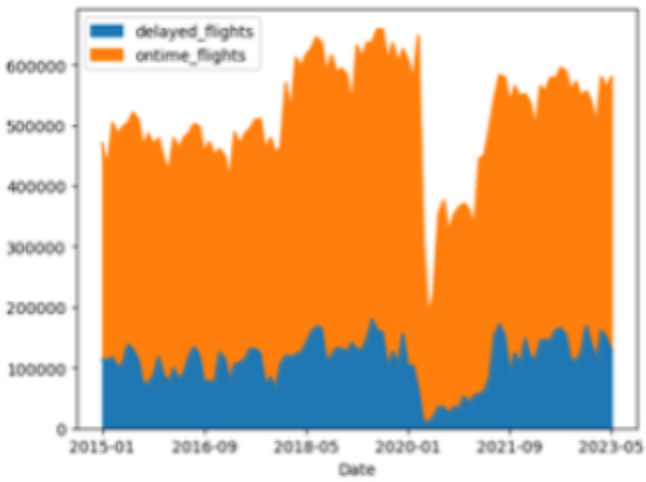


Figure 1: Flight Count Over Time from BTS Data

One issue we needed to immediately address was the validity of data during the height of the COVID-19 pandemic. As shown in Figure 1, the number of flights flown in mid-2020 unsurprisingly dropped precipitously as did the number of flight delays. We decided to focus our work on the data from 2015-2019.

Because the BTS data included the aircraft registration number, we realized we could consider potential features that focused on the characteristics of the specific aircraft. We found that information in the Federal Aviation Administration's (FAA) Releasable Aircraft Database. The FAA releases this database annually as a zip file containing CSV files. For each year, there are individual files including: Aircraft Registration Master File, Aircraft Reference File, Engine Reference File, and Aircraft

Deregistered File. More details about these and other files are provided in the appendix. Also, see [6] for more general information about this dataset.

With BTS flight data back to 2015, we needed to download the current Releasable Aircraft Database as well as each annual release back to 2015. This comprised nine zip files for a total of 586.1 megabytes of data. We needed to combine the information from the Master File with the Aircraft Reference and Engine Reference Files, and then append and deduplicate records from across the years. The dataset was ordered by registration file year descending and deduplicated by N-Number so that the most recent aircraft registration was kept. Once complete, we were able to join the FAA and BTS tables using the aircraft registration number as the join key.

We discovered that some registration numbers were incorrectly recorded in the BTS data, in particular for American Airlines for periods in 2015, 2016, and 2017. Therefore, we were unable to use aircraft information in analyzing delays for that airline.

Lastly, we predicted that the origin or destination airport may have some role in the factor of delay, particularly, where the airport was located relative to others. Northern airports would predictably see more snow and ice in the winter months. We captured these features by joining an additional dataset of airport coordinates by origin and destination airport codes in the original BTS data. This gave us the precise location in relation to other airports, but also allowed the feature to be encoded as a numerical value which was compatible with more automated feature selection and modeling techniques without pre-processing.

Feature Engineering

Our original proposal was to conduct unsupervised feature clustering and principal component analysis to guide the feature selection and engineering we would perform for our supervised learning research. Unfortunately, our time was limited and so we selected and engineered features in parallel for each of our machine learning activities. The reader will find separate feature engineering discussions for supervised learning, time series analysis (part of supervised learning), and unsupervised learning in the sections which follow.

Part A. Supervised Learning

The datasets combined resulted in a 108 feature collection. Feature selection was critical to getting a model to fit with predictive performance, but also simply a reasonable computation time. The initial dataset offered variables ranging from temporal data to granular aircraft information. Our first step was to employ intuitive reasoning to identify a subset of features that were likely to have a significant impact on flight delays. To do this, features were grouped into categories.

- **Temporal Data**: We selected *Month* and *DayOfWeek* as these variables encapsulate the seasonality and weekly patterns that could influence flight delays. The timestamp *FlightDate* was removed based on the intuition that a particular day may have extremely bad weather causing many delays. This would guarantee that our models would overfit to the data since this day will never happen again, it doesn't reveal a particular trend. See "Time Series Prediction" for a full analysis of the temporal nature of this data.
- **Airline and Aircraft Identifiers**: *Reporting_Airline* and *Tail_Number* were chosen to capture the airline-specific and aircraft-specific factors. This also allowed us to filter off by airlines in subsequent model training.
- **Geographical Data**: Airports were encoded in both numeric and categorical form. While *Origin* and *Dest* columns allowed us to easily filter down to select airports, the coordinates of these airports, *Origin_LATITUDE*, *Origin_LONGITUDE*, *Dest_LATITUDE*, and *Dest_LONGITUDE* were included to capture more nuanced geographical patterns, especially concerning northern versus southern airports.
- **Route and Timing**: *CRSDepTime* and *CRSArrTime* were included to represent the planned departure and arrival times from the "Computer Reservation System". The length of the flight was encoded in three variables; *CRSElapsedTime*, *Distance*, and *DistanceGroup*.

- **Aircraft Information**: All of the data from the FAA contributed to these features, including, but not limited to: *year_of_manufacture*, aircraft_model (and *_manufacturer*), engine_model (and *_manufacturer*). We aimed to identify which features might influence the aircraft's reliability and tendency to be delayed.

A correlation analysis was performed to identify features that represented possible duplicate data. For example, The *Mode S Code* is a standardized format transponder code that supports Air Traffic Control. It is unique to each equipped aircraft, so effectively it duplicates the *Tail Number* of the aircraft without providing any additional information to our models. These types of columns were removed from the analysis.

With the intuitively selected features defined in place we could further refine the feature set with automated feature selection. We knew what to look for and we wanted to evaluate the full feature set with both Lasso and Ridge regression techniques to see if they would indicate the same feature coherence. The "delay" label served as our dependent variable for supervised learning. It was generated by taking an OR combination of departure delay and arrival delay, if either are true, we consider the flight "delayed". We then used this dependent variable to run Lasso and Ridge regression. By comparing the ordered feature importance from these two methods and iteratively reducing the list by removing with zero or near-zero coefficients we were able to settle on a compact set of our most predictive features.

By combining intuitive reasoning with automated methods, we were able to distill the feature set to those most relevant for predicting flight delays. This hybrid approach ensured that we did not overlook any potentially impactful variables while also benefiting from the computational rigor of Lasso and Ridge regression. The complete list of features and the notebooks used in this process can be found in the "Jupyter Notebooks & Sample Data" section in the appendix.

Our list of features totaled 15 numerical values and 14 categorical values. Even though some of the categorical features were numeric like "company_type", "engine_type", "aircraft_type", these still represent groups. Most of the categorical values were strings and needed to be transformed to numeric form in order to be processed by our exploratory models. This was achieved by running a ColumnTransformer pipeline on the dataset. Initially, categorical data was transformed by OneHot encoding, but this seemed to increase model complexity and training time. Additionally, OneHot encoding added a variable amount of features depending on how the data was filtered, which may be problematic for follow up testing of a fitted model. For example, Delta uses different aircraft than United, so models that were trained on a subset of data were unable to be cross checked by data from a different airline when using OneHot encoding. Ultimately, numeric features were standardized around zero with the *StandardEncoder*, while we used an *OrdinalEncoder* on the categorical data.

Data was split into training and test sets, then the training set was resampled using the Synthetic Minority Over-sampling Technique (SMOTE). [7] This resampling technique generates synthetic samples in the feature space which rebalances the class distribution and provides a more stable training set for our models. Some of the models (XGBoost and CatBoost) support this type of resampling natively, but the choice was made to train the models on the exact same dataset, which was preprocessed with the exact same random seed during splitting and resampling for consistency in comparison. Our models consisted of probabilistic, non-probabilistic, and tree-based (with ensemble and gradient boosted variants).

- **Logistic Regression:** A probabilistic model that served as our baseline for binary classification.
- **Random Forest:** A tree-based ensemble method that is capable of capturing complex relationships in the data.
- **Support Vector Machine (SVM):** A non-probabilistic binary linear classifier. This model proved to be too computationally intensive to be considered beyond initial training.
- **Histogram-Based Gradient Boosting:** An ensemble method that builds trees in a stage-wise fashion, it is particularly useful for imbalanced datasets like ours.
- **XGBoost:** A highly efficient gradient-boosting library that sequentially builds upon layers of decision trees before compressing into a final model.
- **CatBoost:** A gradient boosted decision tree algorithm (similar to XGBoost) that can handle categorical features natively and proved to be more robust to overfitting.

The models were trained on a subset of data focusing on United Airlines flights that either departed from or landed at O'Hare Airport in Chicago. This subset was chosen for its representativeness and computational efficiency. Each model was run through GridSearchCV with 5-fold cross validation and evaluation metrics were compiled. For efficiency in this round of model selection, *only* the default parameters were passed to each model while targeting the optimal F1 score. Table 1 below includes the cross-validation results. It should be noted here that most models appear to be overfit by the third round of training. This was not well understood at the time, however, in the deep model analysis with CatBoost this was recognized and mitigated as much as possible.

Table 1: Model Selection Cross-Validation Results

Model	Split 1	Split 2	Split 3	Split 4	Split 5	Mean	STD	mean_fit_time
logistic_regression	0.5962	0.6006	0.6091	0.6052	0.6097	0.6042	0.0051	305.54
random_forest	0.4151	0.6366	0.9356	0.9385	0.9379	0.7727	0.2134	81.83
svm	0.5823	0.5777	0.5794	0.5764	0.5783	0.5788	0.0020	2338.47
gradient_boost	0.2263	0.5659	0.9595	0.9574	0.9578	0.7334	0.2956	7.63
xgboost	0.2682	0.5918	0.9590	0.9561	0.9582	0.7466	0.2781	3.84
catboost	0.2599	0.5890	0.9635	0.9616	0.9611	0.7470	0.2832	101.62

Given the imbalanced nature of our dataset, the following metrics (values in Table 2) were chosen to give a comprehensive evaluation of the models on the test set.

- **Accuracy:** Measures the proportion of correct predictions in the total sample.
- **Precision:** Indicates the proportion of true positives among the predicted positives.
- **Recall:** Measures the proportion of true positives among the actual positives.
- **F1-Score:** Harmonic mean of precision and recall, favored due to class imbalance.
- **ROC AUC:** Area under the Receiver Operating Characteristic curve, useful for understanding the trade-offs between true positive rate and false positive rate.

Table 2: Model Selection Test Evaluation Metrics

Model	Accuracy	Precision	Recall	F1	ROC AUC	CV Mean
logistic_regression	0.6013	0.3400	0.5943	0.4325	0.5990	0.6042
random_forest	0.7611	0.5570	0.3219	0.4080	0.6170	0.7727
svm	0.5734	0.3198	0.5929	0.4155	0.5798	0.5788
gradient_boost	0.7800	0.6823	0.2614	0.3780	0.6098	0.7334
xgboost	0.7846	0.6801	0.2975	0.4139	0.6247	0.7466
catboost	0.7869	0.6925	0.2994	0.4180	0.6269	0.7470

Figure 2 below shows the results of model evaluation on the test set. Also included is the mean test score from model training for comparison. Again, this points to the tendency for the tree-based models to overfit given the default parameters. Across the board, the models showed suboptimal performance, indicating that the feature set may lack the predictive power that we initially anticipated.

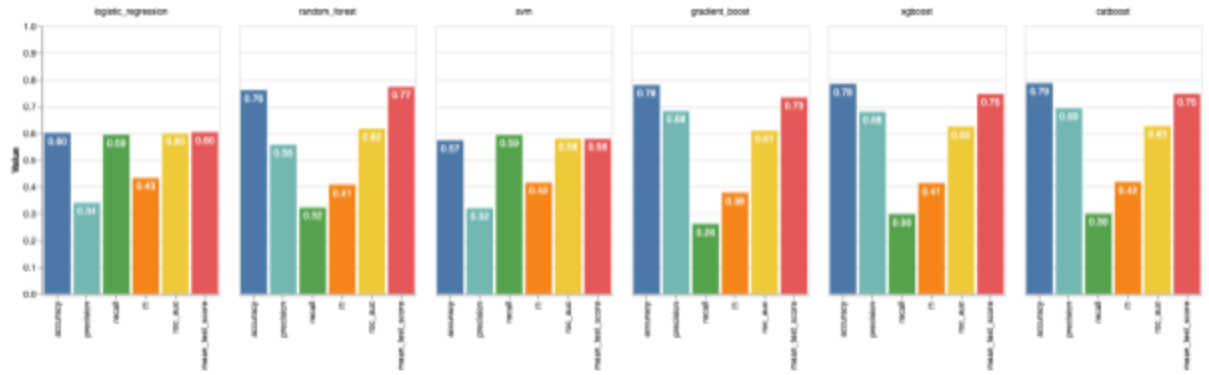


Figure 2: Model Evaluation (All delays, UA, ORD)

CatBoost was chosen for a follow-up in depth analysis. Even though its performance was not highly impressive, it was responsive to hyper-parameter tuning in initial testing and offered a reasonable run time. In order to handle the overfitting problem discovered in the model selection stage the *early_stopping_rounds* parameter was added to the model with the value 20. This enables the model's internal overfitting detector and stops the training after the specified number of iterations since the optimal *Logloss* value. All previous iterations are then collapsed into a single tree. This sped up the execution time and fit a model with consistent performance during both training and test evaluations. Although our intention was to optimize to the F1-score, the CatBoost model does not support this as an optimization metric during training. Log loss considers the prediction probability which makes the model sensitive to how confident it is in its categorical classification. To optimize the hyper-parameters of our CatBoost model a grid search was executed with 3-fold cross validation as well as sample shuffling and stratification to further reduce overfitting. The following parameters proved to be the most responsive in initial testing, the values in bold indicate the best performance in the final model.

- learning_rate: [0.01, 0.1, **0.6**, 1]
- depth: [4, **6**, 8]
- l2_leaf_reg: [**1**, 3, 5]

Metrics were generated to evaluate the performance of the final fitted CatBoost model. (See Figures 3 and 4 below.) Additional visualizations (Precision-Recall Curve and ROC) for this optimized model can be found in the Supervised Model Evaluation section in the appendix.

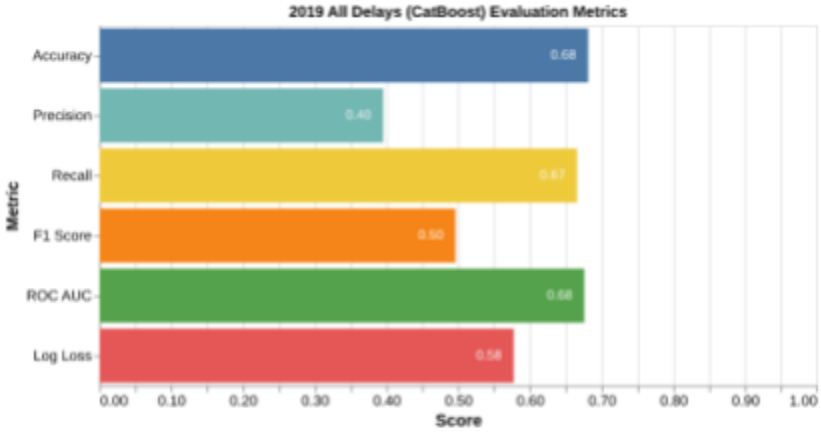


Figure 3: 2019 All Delays (CatBoost) Evaluation Metrics

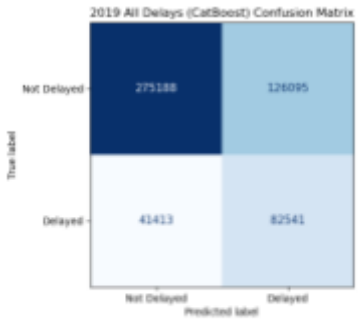


Figure 4: 2019 All Delays (CatBoost) Confusion Matrix

The CatBoost model rated these features as the most important, with the top 4 accounting for over half and the top 10 accounting for nearly 80% of the model's predictive power. Unfortunately, none of these features came from the FAA dataset as we had hoped. It seems that the particular aircraft has a negligible impact on the outcome of a delayed flight. The model was retrained several times with a partial set of features including only the top 15, 9, and 6. Performance very slightly improves with only the top 15 and very slightly declines with only the top 6. The full list of feature importance as well as hyper-parameter sensitivity visualizations can be found in the Supervised Sensitivity Analysis section of the appendix.

Table 3: Supervised Feature Importance (CatBoost)

Rank	Feature	Importance	Cumulative Importance
1	DayofMonth	18.7318	18.7318
2	Month	17.7656	36.4974
3	DayOfWeek	8.4516	44.9490
4	CRSDepTime	7.8024	52.7514
5	Origin_LONGITUDE	5.8846	58.6359
6	Reporting_Airline	5.2053	63.8412
7	Dest_LONGITUDE	4.8329	68.6741
8	CRSArrTime	4.3172	72.9913
9	Origin_LATITUDE	3.6464	76.6377
10	Quarter	2.9810	79.6187

As shown in Table 3 above, the temporal features of *DayofMonth*, *Month*, and *DayOfWeek* rank highly which speaks to the nature of *when* a flight occurs weighing heavily on the outcome of a delay. We conducted a time series analysis to further examine this effect.

Time Series Prediction

We have the advantage of several years of on-time performance data, so it makes sense to see if we can use time series analysis to predict future flight delays. It is not realistic to use tens of millions of individual flight records for this task. We need to aggregate data in such a way that we can apply time series techniques. For this analysis, we aggregated flights to the daily level, initially combining statistics from all reporting airlines.

We want to predict the total number of delayed flights per day over a range of time. For our dataset, a delayed flight is one that has experienced at least a 15-minute delay on departure and/or arrival. This is indicated by the variables *DepDel15* or *ArrDel15* having a value of 1. So, for training and testing, we simply counted those flights.

After exploring the resulting aggregated data, we needed to make two modifications. First, we wanted to avoid complications in the data due to the COVID-19 pandemic. So, in general, we used data from 2015-2018 for training and data from 2019 for testing. (For cross validation, we used an 8-fold sliding window throughout the full time range from 2015 to 2019. Each window used approximately one year of data for training and three months of data immediately following for testing.) We also noticed a discontinuity in the overall trend of total flights in 2018. We discovered that several airlines did not report data prior to that time. Therefore, we eliminated those airlines, sticking to those that had consistent reporting from 2015 through 2019.

Feature engineering was not a major concern for time series analysis. We have a series of delayed flight counts by day, and we simply want to predict those values going forward. However, the models we reviewed can be influenced by using exogenous variables – external information that is not part of the time series but that may have some impact on that series.

There were two different types of exogenous features we considered. First, we looked for interesting information that could be aggregated to the daily level, taking advantage of features that came from both the BTS and FAA data. This eliminated a lot of possible features that did not lend themselves well to aggregation. Ultimately, we chose to explore the following features (which were not relevant for the model with the best performance):

- The total number of flights in a day;
- The total number of aircraft flown in a day;
- The average flight distance in a day;
- The average year of manufacture of the aircraft flown in a day;
- The number of flights on Airbus, Boeing, or Embraer aircraft in a day (3 features); and
- The number of flights with CFM, GE, Pratt & Whitney or Rolls-Royce engines in a day (4 features).

As we will discuss, one of the characteristics of this data is its seasonality. There are patterns one can observe on a weekly, monthly, or annual basis, and these can have a large impact on predictions. Modeling seasonality can be quite computationally intensive, but converting information about the date to a set of exogenous features can have similar influence on the model for a smaller computational cost. We created a series of Fourier terms for different seasonal cycles (weekly, monthly, annual) to discover their impact.

We used these features to support our exploration of **three different methods** for time series analysis: ARIMA/SARIMA/SARIMAX, TBATS, and XGBoost.

Auto-Regressive Integrated Moving Average (ARIMA) has three components:

- Auto-Regression (AR): Predictions are based on the variable's prior values. The hyperparameter *p* is the order of the ARIMA model. It defines how many lags (prior values) are used.
- Moving Average (MA): The regression error is based on a linear combination of errors from a moving window of the current and past observations. The hyperparameter *q* is the order or size of the moving window.
- Integrated (I): If the data is non-stationary (does not have consistent mean and variance), it must be differenced, perhaps several times, to make it so. The hyperparameter *d* is the degree of differencing, or the number of differencing operations needed to induce stationarity.

Seasonal ARIMA (SARIMA) adds a seasonal component identified by the hyperparameters P, D, Q, and m. P, D, and Q are defined similarly to their lowercase components described above, but they apply specifically to the seasonal component of the time series, while m indicates the number of periods in a season. Finally, ***SARIMA with Exogenous Factors (SARIMAX)*** considers exogenous variables provided to the model for fitting and for prediction.

Hyperparameter tuning for ARIMA and SARIMA was performed using the *auto_arima* method from the *pmdarima* Python library. While we had a fixed value for differencing, this method allows the user to specify a range of values for p and q and chooses the best fit based on a chosen criterion. For SARIMAX, we ran several permutations with different combinations of the exogenous variables we identified as part of feature engineering until we found one that minimized error.

TBATS stands for Trigonometric Seasonality, Box-Cox transformation, ARMA Errors, Trend and Seasonal Components. TBATS does not allow for the use of exogenous variables, but it does allow for multiple seasonal patterns, which seem to exist in this dataset. In addition, the hyperparameter for the periodicity of the season need not be an integer, so one can use 365.25 for annual seasonality and account for the influence of the leap year. Hyperparameter tuning was done by selecting combinations of weekly, monthly, and annual seasonality and reviewing the error in the results.

XGBoost is a gradient boosting library using decision trees. While we are using this model as a candidate for binary classification, it can also be applied to time series. This model did not require hyperparameter tuning, as it yielded the best performance "out of the box." However, we did perform a grid search across several hyperparameters and were able to gain a very slight improvement.

We originally intended to simply focus on the ARIMA family of models, as they are very often used for time series prediction in the literature we reviewed. With that said, TBATS and XGBoost appear to offer advantages in accuracy and computational speed that make them worthy of consideration.

Our **time series model evaluation** process, including model building and testing, is fully documented in our notebook entitled "Supervised Learning - Time Series Analysis." The primary metric used to measure performance was the Symmetric Mean Absolute Percentage Error (SMAPE). We had intended to use Mean Absolute Percentage Error (MAPE) as a more explainable metric. However, the Python *pmdarima* package that we used for the ARIMA family uses SMAPE for its cross validation, so we chose to adopt it. SMAPE is generally a better measure when data contains many zero values. That is not the case here, so SMAPE slightly

underestimates the error when compared to MAPE. We also included Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for comparison, as these are well-known metrics.

Based on MAE, SMAPE, and RMSE, XGBoost provided the best cross-validated results. Table 4 below shows how the various models compared (standard deviations are in parentheses):

Table 4: Cross Validation Results for Time Series Prediction

Model	SMAPE	MAE	RMSE
ARIMA(3,1,1)	35.7% (5.8%)	1319.4 (259.8)	1565.1 (293.2)
SARIMA(0,1,2)(1,0,1,30)	35.1% (5.8%)	1279.6 (207.3)	1503.5 (219.7)
SARIMAX(0,1,3)(*)	37.4% (7.6%)	1391.8 (291.5)	1629.7 (288.7)
SARIMAX(1,1,1)(+)	34.7% (5.1%)	962.8 (170.8)	1129.5 (192.3)
TBATS(7, 365.25)	34.2% (7.1%)	945.0 (237.3)	1111.4 (233.3)
XGBoost	**23.7% (3.1%)**	**610.8 (79.8)**	**784.8 (98.4)**

In the table above, (*) indicates that the SARIMAX model was not formally trained on the seasonal parameters. Instead, we created Fourier features that simulated annual seasonality. Also, (+) indicates that seasonality was not formally trained. However, for this model, we used the Fourier features as well as exogenous variables from the original data.

For model selection in the ARIMA family, we relied on the *auto_arima* method from the *pmdarima* library. Given a set of training data and parameter constraints, this method will perform the equivalent of a grid search to find the best fit. There are several choices of metric for best fit, the most common being Akaike Information Criterion (AIC). This produces useful models, but may not minimize SMAPE, MAE, or RMSE, although we certainly measured those metrics for each model we trained and tested.

Once a model was selected, we ran it through an eight-fold sliding window cross validation. Each window contained approximately one year of training data and three months of test data. At each fold, the window moved forward approximately six months. Each window moves forward in time to avoid data leakage into the model. Our cross validated metrics were never as good as individual training runs, but this is not surprising. The amount and nature of the training and test data in each fold has an impact on the overall mean result.

It was quickly clear that ignoring the seasonality of the data with a basic ARIMA model was not producing good results. The best model, ARIMA(3,1,1), had a SMAPE of 35.7%. So we quickly moved on to SARIMA. For SARIMA, one of the biggest challenges turned out to be identifying the correct seasonality of the data. We suspected we could have weekly, monthly, and/or annual seasonality. SARIMA generally can only model one seasonality pattern at a time. The longer the season, the greater the computational need. It became quickly clear that we could not train models with annual seasonality given our computing constraints.

We attempted training models in the standard fashion for weekly and monthly seasonalities. While a model with monthly seasonality showed improvement over the ARIMA model, the improvement was small and the model did not seem to capture the variance in the data. (SARIMA(0,1,2)(1,0,1,30) had a SMAPE of 35.1%.)

This is where we were able to employ a trick to model longer seasonal patterns. We could create Fourier features using a method within the *pmdarima* library. The method generates an array of exogenous features that uses sine and cosine terms to emulate the seasonal pattern. This also allows us to generate multiple seasonal patterns if the data calls for it, which it does in this case. Fitting a SARIMAX model in this fashion yielded a result that “looked” more correct. However, the cross validation metrics were not as promising.

We continued with SARIMAX and added in the 11 exogenous features described earlier in this section, along with the Fourier features to continue to model seasonality. This finally yielded a model with better performance than our previous SARIMA attempts. (This SARIMAX(1,1,1) had a SMAPE of 34.7%.) With more time, we might have chosen different combinations of the exogenous features to see if we could find additional improvement. However, there were more performant models waiting for us.

For TBATS, we trained models using a few different combinations of weekly, monthly, and annual seasonalities. Using weekly and annual - TBATS(7, 365.25) - offered the best result. SMAPE for this model dropped to 34.2%.

In addition to using XGBoost for our unsupervised learning and our binary classification, it turns out that this model can be very useful for time series forecasting, as well. We needed to create features that modeled various components of the date. Using a suggestion from [13], we extracted the day of week, quarter, month, year, day of year, and day of month. Figure 5 below shows the feature importance in the XGBoost model.

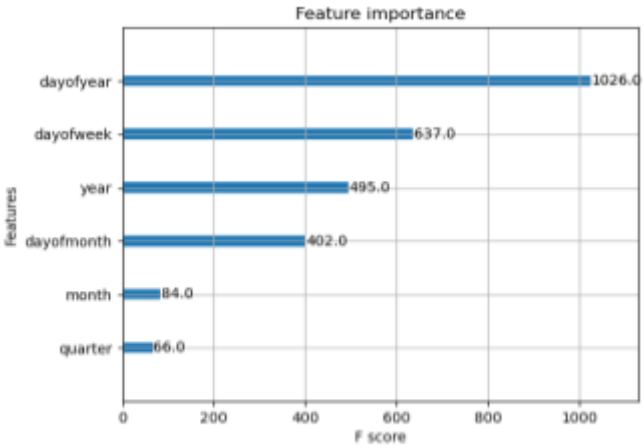


Figure 5: Time Series Feature Importance for XGBoost

We were quite pleased to find that day of year and day of week were the most important features. This lends credence to the assumption that both weekly and annual seasonality were exhibited in our dataset.

Using XGBoost with no further tuning yielded the **best results of any model**, with a SMAPE of 23.8%. We did look at hyperparameter sensitivity by performing a grid search of the hyperparameters shown in Figure 6 below:

```
params = {
    'learning_rate':[0.1, 0.3, 0.5, 0.7],
    'gamma':[0, 0.5, 1.0, 1.5, 2.0],
    'max_depth':[2, 4, 6, 8],
    'min_child_weight':[1, 3, 5],
    'subsample':[0.25, 0.5, 0.75, 1.0],
    'colsample_bytree':[0.25, 0.5, 0.75, 1.0],
    'colsample_bylevel':[0.25, 0.5, 0.75, 1.0],
    'colsample_bynode':[0.25, 0.5, 0.75, 1.0],
}
```

```
{'colsample_bylevel': 0.75,
 'colsample_bynode': 0.75,
 'colsample_bytree': 0.75,
 'gamma': 0,
 'learning_rate': 0.3,
 'max_depth': 8,
 'min_child_weight': 1,
 'subsample': 0.75}
```

Figure 6: Parameters Chosen for XGBoost Grid Search and Results

The grid search took more than an hour to complete, and only improved the SMAPE by 0.1%. Given the standard deviation, this seems an insignificant improvement. So we conclude that, at least for this dataset, the XGBoost model is relatively insensitive to hyperparameter changes. Finally, we used the tuned XGBoost model to predict delayed flights for several airlines. As shown in Table 5 below, no single airline performed better than the aggregated dataset, with increases in SMAPE ranging from 4.8% to 15.4%. We have been able to discern no specific reason for this performance, other than recognizing that we are necessarily using less training and test data for an individual airline than we are for the aggregated set.

Figure 7 below shows the progression of our time series models. Each figure within shows the best single training run for that particular model. Blue represents the training data, green represents the test data, and orange indicates the model predictions. Reading from left to right and top to bottom, the figure includes: ARIMA, SARIMA, SARIMAX (with Fourier features), SARIMAX (with Fourier and exogenous features), TBATS, and XGBoost (tuned). (Also see Appendix G.)

Table 5: Time SeriesResults by Airline

Airline	SMAPE (Change)
American (AA)	28.5% (+4.8%)
Alaska (AS)	32.3% (+8.6%)
Delta (DL)	38.7% (+15.0%)
United (UA)	33.2% (+9.5%)
Southwest (WN)	29.8% (+6.1%)
Frontier (F9)	39.1% (+15.4%)
JetBlue (B6)	35.4% (+11.7%)
Spirit (NK)	36.3% (+12.6%)

Tradeoffs we discovered during our exploration of time series models included performance of a single training run when compared to cross validation. Our initial training runs always used four years of training data and one year of test data. The cross validation suggests that such a ratio may be contributing to overfitting. Also, we had to aggregate the data set in order to make it useful for time series modeling. This may have diluted the power of certain features and may have contributed to our relative poor overall performance.

As for **failure analysis**, the largest concern is that the mean error, even from our most highly performant model, is simply too high. We could not recommend that anyone use such a model to make a travel decision or to guide policy. It seems likely that there are unobserved variables that may substantially influence flight delays, including weather, labor conditions, holiday anomalies, ticket prices, or cost of consumables (oil, fuel, etc.). We also discovered that the ARIMA family of models can be very finicky. Even though we were selecting models that explicitly model seasonality, several combinations seemed to ignore that pattern. Even improved models using exogenous variables did not show substantially better performance. We were fortunate to have chosen other possibilities like TBATS and XGBoost. Finally, as mentioned earlier, our chosen model is not useful for predictions at the airline level. There may be specific policies followed by a given airline that have larger influence over delays than what we can observe in this dataset. Overall, it would be useful to identify specifics about some of these unobserved influences and see if we could find relevant information to enhance our current dataset.

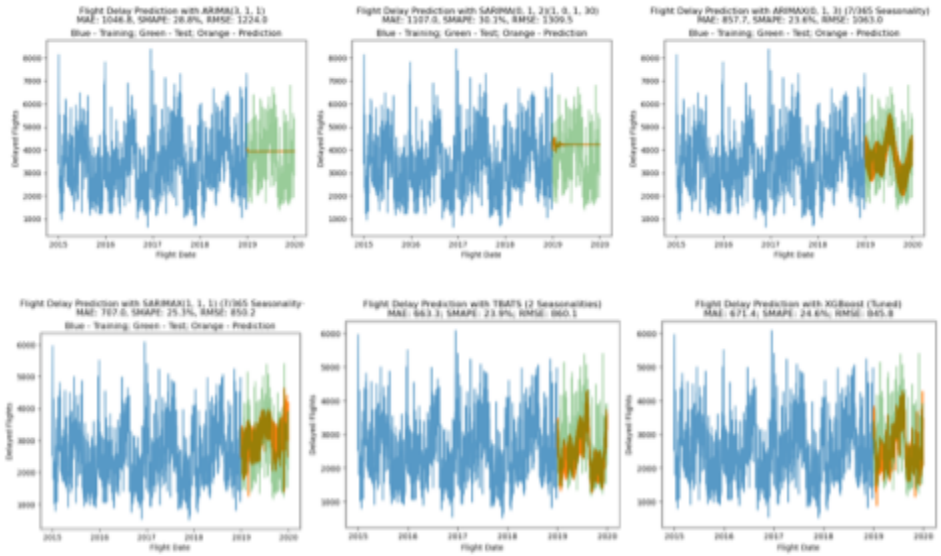


Figure 7: Progression of Time Series Prediction Models

Part B. Unsupervised Learning

Our dataset consists of multiple categorical columns, and due to the categorical nature of the data, selecting an appropriate method for effective feature handling is crucial. To address this challenge, we employed an Automated Feature Selection approach [8], utilizing the Python library AutoFeatSelect. This library is specifically designed to automate and expedite the feature selection process in machine learning projects. We conducted a feature importance analysis [9] using two different algorithms: XGBoost and Random Forest. Subsequently, we combined the results and identified overlapping features from both analyses. To validate the selected features, we applied the dataset with the selected features only to a range of machine learning models using LazyClassifier, a straightforward and efficient tool for making predictions. Among these models, three demonstrated superior performance, achieving an average accuracy of 92%, balanced accuracy of 85%, ROC AUC of 85%, and an F1 Score of 92%. Furthermore, we applied PCA to reduce dimensionality to four components, with approximately 80% cumulative explained variance.

The final set of selected features includes:

- **ArrTimeBlk**: CRS Arrival Time Block, Hourly Intervals. This variable is categorical.
- **TaxiOut**: represents the duration, in minutes, of the aircraft's ground movement before taking off. It is an integer.
- **WheelsOn**: Wheels On Time. This variable is categorical.
- **DepTimeBlk**: CRS Departure Time Block, Hourly Intervals. This variable is categorical.
- **WheelsOff**: Wheels Off Time. This variable is categorical.

- **MODE S CODE**: It represents Aircraft Transponder Code. This variable is categorical. This code is assigned by air traffic control when the aircraft operates within controlled airspace. Its primary purpose is to assist ATC in identifying and tracking the aircraft's position on radar screens. This variable is categorical.
- **TaxiIn**: represents the duration, in minutes, of the aircraft's ground movement after landing. This variable is integer.
- **CRSElapsedTime**: CRS Elapsed Time of Flight. This variable is integer.

From the results, it's evident that flight delays are closely related to airport operations, including factors such as ground congestion, peak operational times during the day, and the planned flight duration. This aligns with the results of the exploratory analysis on delay types, which indicates that late aircraft and carrier-related issues are the major causes of delays for the airlines, and these are closely related to airport operations. Graphical representation of delay causes is available in Appendix E.

The Python library *AutofeatSelect* is a handy tool for feature selection as it helps with time and resource saving while improving model performance. It streamlines and expedites the feature selection process in machine learning projects by calculating feature importance scores, using various ranking methods, and aiding in the identification and elimination of highly correlated variables. Here's an outline of how we implemented this technique:

- **Data Preparation**: In our data preparation process, we focused on 2019 flight records for all major airlines that were neither canceled nor diverted. Initially, we conducted a preliminary filtering of columns strongly associated with arrival and departure delay labels. Subsequently, we categorized the remaining columns into two types: categorical and numeric. It's worth noting that in our dataset, several categorical columns are represented in numerical form, including time and model codes etc. To address missing values in categorical columns, we filled them with an empty string, while numerical columns did not contain any null values. Finally, we added the delay label based on the labels of arrival delay and departure delay, considering delays of over 15 minutes.
- **Data Split**: We divided the dataset into three distinct segments: the training set, the testing set, and the validation set. Given that airlines operate under different business models and cater to varying target customer groups, our data split was stratified by airline, ensuring that each segment appropriately represents the diverse airline profiles within the dataset.
- **Correlation Analysis**: We identified highly correlated features using the numeric_correlations and *categorical_correlations* methods of the *CorrelationCalculator* class. These features were subsequently removed from both the numerical and categorical features lists. Our approach utilized Pearson correlation coefficients, and we set a threshold of 0.8 for the correlation analysis.
- **Feature Selector Object**: Following the elimination of correlated features from the initial feature set, our next step involved the creation of a feature selector object. In alignment with the project's objective of predicting a binary delay label, we chose a "classification" modeling approach. The feature selector was trained using both the training and test datasets, while the validation dataset was set aside for later model validation purposes.
- **Feature Selection Methods**: We computed feature importance scores using both the XGBoost and Random Forest algorithms, leveraging the 'n_jobs' parameter to expedite the process. While Boruta is another effective option for feature selection, it was deemed computationally expensive for this case. Nevertheless, it demonstrated its efficacy in later sensitivity validation tests when it worked well on a smaller dataset.

XGBoost: Given the massive size of our dataset, XGBoost stands out as an efficient choice. It is renowned for its ability to expedite both training and execution processes. Graphical representation of XGBoost Importance is available in Appendix F.

Random Forest: With categorical variables playing a pivotal role in our dataset, Random Forest is an ideal candidate. It can natively handle categorical data, eliminating the need for extensive preprocessing like one-hot

encoding. This simplifies the feature selection process and can yield more efficient and interpretable results. Graphical representation of Random Forest Importance is available in Appendix F.

Boruta: Boruta is an extension of the Random Forest algorithm, offering a comprehensive approach to feature selection. It compares features against a randomized dataset to enhance the accuracy of the selection process by ranking features based on their importance and statistical significance. However, it comes with significant computational demands, making it less suitable for our large dataset. Nevertheless, it has demonstrated its effectiveness during sensitivity testing on a smaller dataset, specifically with Delta Airlines' 2019 data.

Final Feature Selections: We extracted the top 10 most important features from each set of feature importance results. (See Table 6.) By performing an inner join among these selected features, we identified 8 common features that appeared in all importance rankings.

Table 6: Final Feature Selections

XGBoost		Random Forest	
Feature and Importance		**Feature and Importance**	
ArrTimeBlk	0.188	WheelsOn	0.143
TaxiOut	0.139	WheelsOff	0.122
WheelsOn	0.131	ArrTimeBlk	0.087
DepTimeBlk	0.121	TaxiOut	0.086
WheelsOff	0.094	DepTimeBlk	0.065
MODE S CODE	0.064	ActualElapsedTime	0.054
TaxiIn	0.048	MODE S CODE	0.053
Month	0.047	TaxiIn	0.042
DayofMonth	0.03	Distance	0.042
CRSElapsedTime	0.019	CRSElapsedTime	0.04
ActualElapsedTime	0.018	DayofMonth	0.036
DayOfWeek	0.017	Origin_LONGITUDE	0.031
NO-SEATS	0.016	Dest_LONGITUDE	0.031
Distance	0.015	Origin_LATITUDE	0.029
Origin_LONGITUDE	0.012	Dest_LATITUDE	0.029
Dest_LONGITUDE	0.01	Month	0.028
Origin_LATITUDE	0.01	NO-SEATS	0.023
DistanceGroup	0.009	DayOfWeek	0.022
Dest_LATITUDE	0.006	THRUST	0.021
THRUST	0.005	DistanceGroup	0.012
HORSEPOWER	0	HORSEPOWER	0.005
SPEED	0	NO-ENG	0
NO-ENG	0	SPEED	0

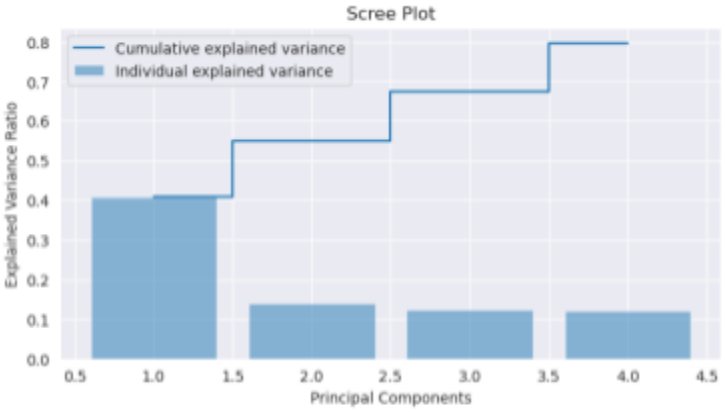


Figure 8: PCA Scree Plot

PCA: We applied Principal Component Analysis (PCA) to reduce the dimensionality of the dataset from 8 dimensions to 4. Using these four components, we achieved an 80% cumulative explained variance, which is often sufficient for many supervised learning models. (See Figure 8.) It is worth noting that we assumed the dataset is normally distributed due to the size of the dataset, based on the Central Limit Theorem (CLT).

To validate the results of our feature selection process, we employed LazyClassifier from Lazy Predict [10]. LazyClassifier streamlines the model selection process, eliminating the need for extensive parameter tuning. We found it to be a valuable tool for assessing the efficiency of the selected features in various machine learning models. Before deploying LazyClassifier, we performed categorical-to-numerical encoding by applying LabelEncoder(). Subsequently, we carefully selected a subset of model classifiers [11][12] that are well-suited and practical for our dataset:

LogisticRegression: LogisticRegression is inherently tailored for binary classification tasks. It adeptly handles categorical variables, simplifying the modeling process. This model can serve as an effective baseline for

comparison due to its straightforward nature. Additionally, it is expected to be computationally efficient, which is beneficial for our massive dataset.

RandomForestClassifier: The feature selection results stem from the Random Forest classifier. We anticipated that this model would deliver improved performance based on the feature selection process.

XGBClassifier: Similar to RandomForestClassifier, the feature selection outcomes originate from the XGBClassifier. We have high expectations for this model's performance, given its feature selection and predictive capabilities.

LGBMClassifier: Similar to LogisticRegression, it is designed to handle large datasets. It offers faster training and prediction times compared to several other algorithms, making it well-suited for your dataset's size and complexity. LGBMClassifier naturally accommodates categorical variables, streamlining the modeling process.

With the data preparation phase finalized, the next step is to build a LazyClassifier model. This model is trained on both the training dataset and the validation dataset. The outcomes are in accordance with our expectations. Random Forest Classifier and XGB Classifier exhibit the highest performance, with LGBM Classifier following closely. Notably, Logistic Regression stands out for its efficiency in delivering results within the shortest timeframe.

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
RandomForestClassifier	0.93	0.86	0.86	0.93	2209.91
XGBClassifier	0.92	0.85	0.85	0.92	23.69
LGBMClassifier	0.91	0.83	0.83	0.91	28.36
LogisticRegression	0.81	0.59	0.59	0.76	7.36

Figure 9: Validation Result - All Major Airlines

The above results effectively demonstrated the impact of the feature selection process. We further tested its performance on the 2019 data of a specific airline, Delta Airlines, which yielded favorable results as follows:

During this phase, we implemented the Boruta ranking method, which revealed that six features with the top 2 rankings overlapped with the final selected features. Surprisingly, 'DepTimeBlk' and 'ArrTimeBlk,' which were categorical columns, were removed during the correlation analysis. Furthermore, it's important to acknowledge that some features within the final selection could be sensitive to specific airlines. This observation aligns with the fact that airlines often operate with distinct procedures, cater to diverse customer bases, and follow unique flight patterns. As a result, the relationships between features and delay labels can vary significantly. Features that prove crucial for predicting delays across all airlines may lose relevance when applied to a specific airline. Conversely, airline-specific features that are essential for predicting delays on a specific carrier might not emerge during the feature selection process when analyzing a combined dataset.

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
RandomForestClassifier	0.94	0.86	0.86	0.94	275.39
XGBClassifier	0.93	0.84	0.84	0.93	4.02
LGBMClassifier	0.92	0.81	0.81	0.92	4.57
LogisticRegression	0.84	0.58	0.58	0.80	1.20

Figure 10: Validation Result - Delta Airlines

Discussion

While we did implement a stratified split by airlines during feature selection, it's important to note that the resulting feature selection may not be universally applicable to all airlines. When applying the Boruta method, we observed that two features were filtered out early in the feature selection phase. Given more time, it would be beneficial to conduct feature selection individually for each airline to identify features that are more airline specific. We also

wonder if the results would differ if the feature selector were executed on data spanning multiple years. This is an avenue we would explore if we had the luxury of additional time and computing resources.

The size of the dataset is one of the challenges we have encountered. We made the deliberate choice to concentrate on one year's worth of data for major airlines and exclude data beyond 2019 for the unsupervised learning. This decision was influenced by the significant impact of the pandemic on the aviation industry, which introduced a substantial deviation in the data trends. It looks like there's a minor typo in the sentence. Without the time constraint, it would be interesting to see what current flight trends look like after the WHO announced that COVID is no longer a global threat. To tackle high memory consumption, we attempted to promptly delete variables with large memory footprints when they were no longer in use to free up memory. Additionally, adjusting the n_jobs parameter significantly improved computation speed.

The feature selection outcome includes just one feature from the FAA dataset. This outcome suggests that the features within the FAA dataset do not appear to have substantial significance in predicting the delay label, contrary to our initial expectations. Given our supervised learning analysis, none of the FAA data was predictive of flight delays. In unsupervised and time-series analysis, only a few FAA features were weakly predictive. The time series analysis showed that seasonality is a factor and if given more time we would have liked to incorporate weather data and focus on this as a specific type of delay. Presumably, a narrower problem might have presented an easier target for our supervised models to hit.

Despite the large volume of data we collected, our models experienced errors in time series prediction large enough that they should not be used in production environments. In the future, it might be enlightening to identify the sources of currently unobserved influence and see if they can be quantified and added to our models.

Ethical Considerations

There is a limited amount of personal data that was included in this project. While the FAA dataset does contain personal home addresses for privately-owned aircraft this was not included in any of the modeling work.

Our supervised model focused on data from the top 6 airlines and the top 10 destination airports. While this approach provided a manageable dataset that we felt was representative of the problem we were trying to solve, it likely shows how we introduced selection bias into the project. The selected airlines and airports are among the largest in the United States and may not be representative of the overall airline industry, particularly smaller regional carriers and airports. Each airline operates with distinct business models, caters to diverse customer bases, and follows unique operational procedures. Therefore, it's crucial to recognize that the results may not directly apply to airlines with different characteristics.

Lastly, given the limited predictive power of our models, we strongly advise against using these models as the sole basis for making travel decisions. The models are not sufficiently reliable to predict flight delays with high confidence.

Statement of Work

Aaron Newman	Cecilia Chen	Patrick Sollars
BTS dataset unification and cleaning; Time-series modeling	BTS exploratory data analysis; Unsupervised feature selection and modeling	FAA dataset unification, cleaning and EDA; Supervised feature selection and modeling; GitHub repo setup

All Team Members: Selecting and troubleshooting shared computing environment; Report writing and editing

References

[1] Packer, Adam and Aaron Newman, “Measuring the Continuing Impact of the COVID-19 Pandemic on Domestic Air Travel,” *University of Michigan SIADS 593 Winter 2023 Final Project*. https://github.com/newmanar/SIADS593. (Accessed 15 October 2023.)

[2] Fabien, Daniel, “Predicting flight delays [Tutorial],” *Kaggle*, 2017. https://www.kaggle.com/code/fabiendaniel/predicting-flight-delays-tutorial. (Accessed 1 September 2023.)

[3] Prabakaran N. and Rajendran Kannadasan, “Airline Delay Predictions using Supervised Machine Learning,” *International Journal of Pure and Applied Mathematics*, 119(7), February 2018. https://www.researchgate.net/publication/325034541_Airline_Delay_Predictions_using_Supervised_Machine_Learning. (Accessed 1 September 2023.)

[4] Herbas, Javier, “Using Machine Learning to Predict Flight Delays,” *Analytics Vidhya*, 17 October 2020. https://medium.com/analytics-vidhya/using-machine-learning-to-predict-flight-delays-e8a50b0bb64c. (Accessed 1 September 2023.)

[5] Bureau of Transportation Statistics (BTS), “Airline On-Time Performance Data,” *BTS TranStats Website,* https://www.transtats.bts.gov/Tables.asp?QO_VQ=EFD&QO_anzr=Nv4yv0r%FDb0-gvzr%FDcr4s14zn0pr%FDQn6n&QO_fu146_anzr=b0-gvzr. (Accessed 15 October 2023.)

[6] Federal Aviation Administration, “Aircraft Registration: Releasable Aircraft Database Download,” *Federal Aviation Administration Website*, https://www.faa.gov/licenses_certificates/aircraft_certification/aircraft_registry/releasable_aircraft_download. (Accessed 15 October 2023.)

[7] N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” Journal of artificial intelligence research, 321-357, 2002. https://www.jair.org/index.php/jair/article/view/10302 (Accessed 21 October 2023.)

[8] Canga, Doruk “Automated Feature Selection for Machine Learning in Python” https://python.plainenglish.io/automated-feature-selection-for-machine-learning-in-python-2ad4bcfac19a (Accessed 21 October 2023.)

[9] “Automated Feature Selection & Importance” *Library Documentation* https://github.com/dorukcanga/AutoFeatSelect/blob/main/autofeatselect/auto_feat_select.py (Accessed 21 October 2023.)

[10] “Horse” https://www.kaggle.com/code/huseyinbaytar/horse (Accessed 21 October 2023.)

[11] Loukas, Serafeim “How To Use The LazyPredict Python Library To Select The Best Machine Learning Model In One Line” https://pub.towardsai.net/how-to-use-the-lazypredict-python-library-to-select-the-best-machine-learning-model-in-one-line-4c9e730058b (Accessed 21 October 2023.)

[12] “Lazy Predict” *Library Documentation* https://github.com/shankarpandala/lazypredict/blob/dev/lazypredict/Supervised.py#L15 (Accessed 21 October 2023.)

[13] Mulla, Rob, “[Tutorial] Time Series forecasting with XGBoost,” *Kaggle websit*e. https://www.kaggle.com/code/robikscube/tutorial-time-series-forecasting-with-xgboost. (Accessed 22 October 2023.)

List of Appendices

Appendix A: Jupyter Notebooks & Sample Data

- https://github.com/psollars/milestone-2

As directed, we are including copies of our Jupyter notebooks as well as sample data as part of the zip file to be submitted via Coursera. This document, as well as our notebooks are also available in the GitHub repository listed above.

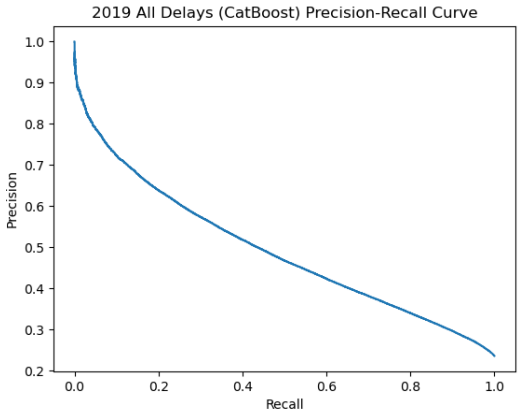
Data samples included in our Coursera submission are all samples from a prepared join of BTS and FAA data, as follows:

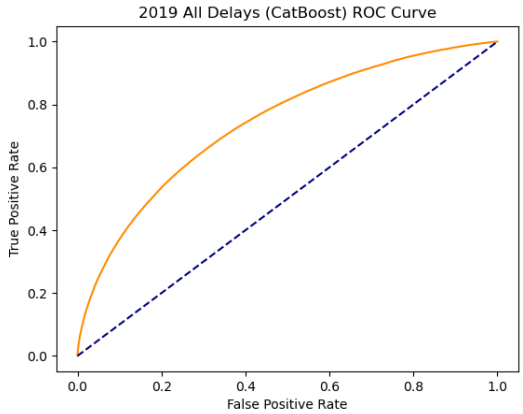
- all_features_2019_sample.csv (used for supervised learning)
- bts_faa_coords_sample.csv (used for unsupervised learning and time series analysis)
- top_airline_airport_2019_sample.csv (used for supervised learning - filtered)
- top_airline_airport_consolidated_features_2019.csv (used for supervised learning - filtered)

Notebooks in our Coursera submission include:

- data_samples
    - data_sampling.ipynb
    - bts_faa_coords_sample.csv
    - all_features_2019_sample.csv
    - top_airline_airport_2019_sample.csv
    - top_airline_airport_consolidated_features_2019.csv
- data_cleaning
    - faa_join
    - Build Clean Sqlite
    - bts_faa_join
    - airport_codes
- eda
    - delay_type_analysis
    - top_airports_2019
    - EDA for Flight Delays Prediction
    - faa_eda
    - bts_eda
- feature_selection
    - Feature Selection Sensitivity Testing
    - feature_selection_2019
    - consolidated_all_delays_feature_selection
- supervised
    - model_selection_all_delays_ORD_UA_logistic_regression
    - model_selection_all_delays_ORD_UA_random_forest
    - model_selection_all_delays_ORD_UA_svm
    - model_selection_all_delays_ORD_UA_gradient_boost
    - model_selection_all_delays_ORD_UA_xgboost
    - model_selection_all_delays_ORD_UA_catboost
    - catboost_grid_search
    - catboost_sensitivity_analysis
- time_series
    - Supervised Learning - Time Series Analysis (including notebook and HTML)
- unsupervised
    - Unsupervised Learning for Flight Delays Prediction
- visualizations
    - Any visualizations that were generated from the notebooks above

Appendix B: Supervised Model Evaluation

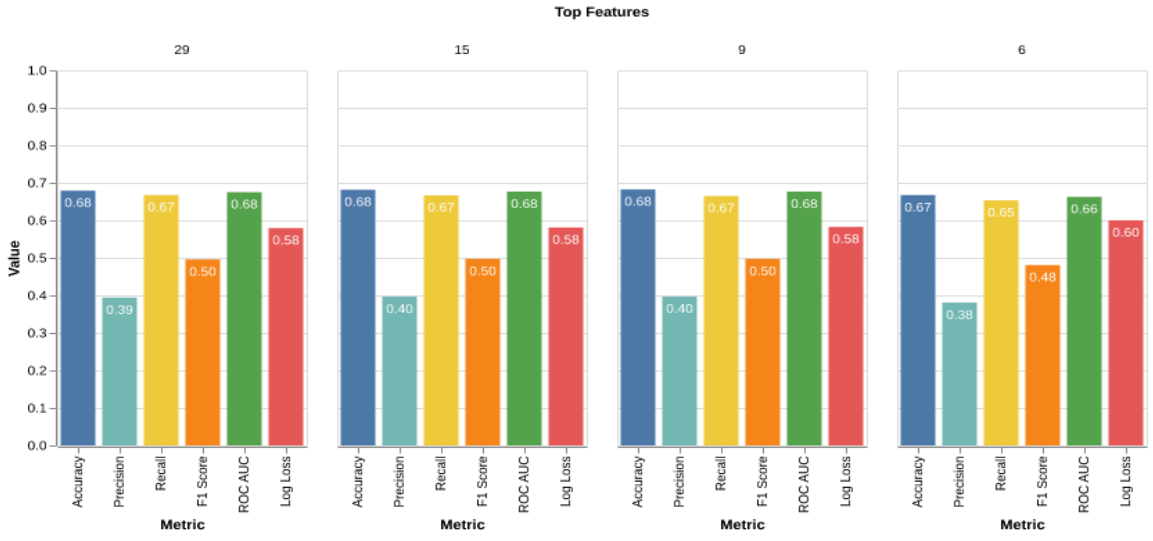


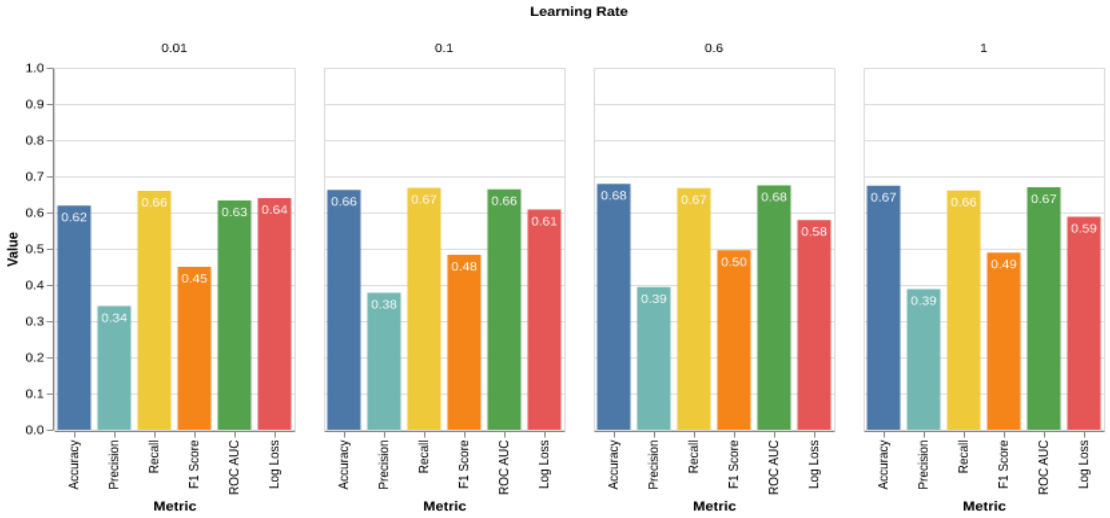


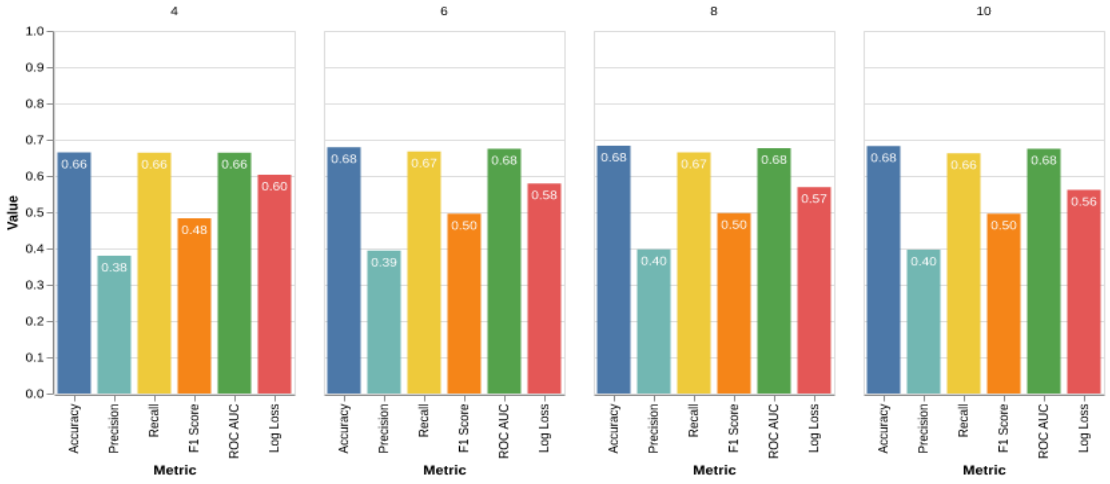
Complete feature importance from CatBoost model trained on 2019 flight data from 6 top airlines and 10 top destination airports.

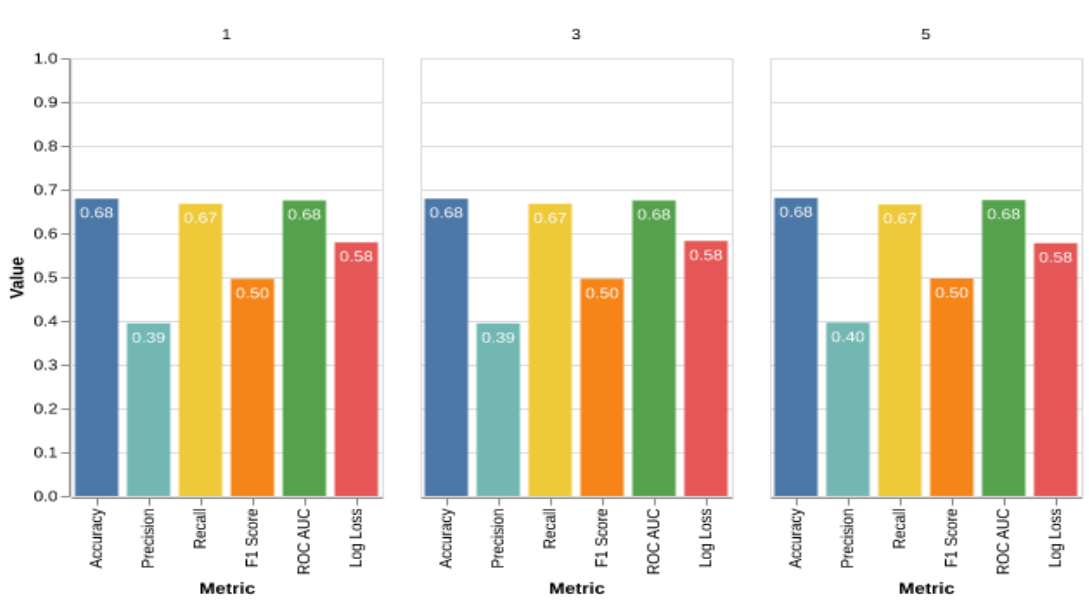
Rank	Feature	Importance	Cumulative Importance	Rank	Feature	Importance	Cumulative Importance
1	DayofMonth	18.7318	18.7318	**16**	Tail_Number	1.9868	93.6637
2	Month	17.7656	36.4974	**17**	aircraft_model	1.5336	95.1973
3	DayOfWeek	8.4516	44.9490	**18**	year_of_manufacture	1.4117	96.6090
4	CRSDepTime	7.8024	52.7514	**19**	engine_model	1.1736	97.7826
5	Origin_LONGITUDE	5.8846	58.6359	**20**	registrant	0.9765	98.7592
6	Reporting_Airline	5.2053	63.8412	**21**	num_seats	0.3142	99.0734
7	Dest_LONGITUDE	4.8329	68.6741	**22**	engine_manufacturer	0.2106	99.2840
8	CRSArrTime	4.3172	72.9913	**23**	aircraft_manufacturer	0.1770	99.4610
9	Origin_LATITUDE	3.6464	76.6377	**24**	aircraft_type	0.1556	99.6167
10	Quarter	2.9810	79.6187	**25**	DistanceGroup	0.1319	99.7486
11	CRSElapsedTime	2.7767	82.3954	**26**	registration_status	0.0704	99.8190
12	Dest_LATITUDE	2.4248	84.8202	**27**	engine_type	0.0669	99.8859
13	Origin	2.4247	87.2449	**28**	company_type	0.0573	99.9432
14	Distance	2.3765	89.6214	**29**	aircraft_usage	0.0568	100.0000
15	Dest	2.0555	91.6769				

Supervised Hyper-Parameter and Feature Sensitivity Analysis




Learning Rate
0.01
0.1
0.6
1
Value
Metric
Accuracy
Precision
Recall
F1 Score
ROC AUC
Log Loss
0.62 0.34 0.66 0.45 0.63 0.64
0.66 0.38 0.67 0.48 0.66 0.61
0.68 0.39 0.67 0.50 0.68 0.58
0.67 0.39 0.66 0.49 0.67 0.59


Depth
4
6
8
10
Value
Metric
Accuracy
Precision
Recall
F1 Score
ROC AUC
Log Loss
0.66 0.38 0.66 0.48 0.66 0.60
0.68 0.39 0.67 0.50 0.68 0.58
0.68 0.40 0.67 0.50 0.68 0.57
0.68 0.40 0.66 0.50 0.68 0.56


L2 Leaf Regularization
1
3
5
Value
Metric
Accuracy
Precision
Recall
F1 Score
ROC AUC
Log Loss
0.68 0.39 0.67 0.50 0.68 0.58
0.68 0.39 0.67 0.50 0.68 0.58
0.68 0.40 0.67 0.50 0.68 0.58

Appendix C: All Available Features from BTS On-Time Performance Data

Field Name	Description
Year	Year
Quarter	Quarter (1-4)
Month	Month
DayofMonth	Day of Month
DayOfWeek	Day of Week
FlightDate	Flight Date (yyyymmdd)
Reporting_Airline	Unique Carrier Code. When the same code has been used by multiple carriers, a numeric suffix is used for earlier users, for example, PA, PA(1), PA(2). Use this field for analysis across a range of years.
DOT_ID_Reporting_Airline	An identification number assigned by US DOT to identify a unique airline (carrier). A unique airline (carrier) is defined as one holding and reporting under the same DOT certificate regardless of its Code, Name, or holding company/corporation.
IATA_CODE_Reporting_Airline	Code assigned by IATA and commonly used to identify a carrier. As the same code may have been assigned to different carriers over time, the code is not always unique. For analysis, use the Unique Carrier Code.
Tail_Number	Tail Number
Flight_Number_Reporting_Airline	Flight Number
OriginAirportID	Origin Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.
OriginAirportSeqID	Origin Airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time.
OriginCityMarketID	Origin Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.
Origin	Origin Airport
OriginCityName	Origin Airport, City Name
OriginState	Origin Airport, State Code
OriginStateFips	Origin Airport, State Fips
OriginStateName	Origin Airport, State Name
OriginWac	Origin Airport, World Area Code
DestAirportID	Destination Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.
DestAirportSeqID	Destination Airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time.
DestCityMarketID	Destination Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.
Dest	Destination Airport
DestCityName	Destination Airport, City Name
DestState	Destination Airport, State Code

Field Name	Description
DestStateFips	Destination Airport, State Fips
DestStateName	Destination Airport, State Name
DestWac	Destination Airport, World Area Code
CRSDepTime	CRS Departure Time (local time: hhmm)
DepTime	Actual Departure Time (local time: hhmm)
DepDelay	Difference in minutes between scheduled and actual departure time. Early departures show negative numbers.
DepDelayMinutes	Difference in minutes between scheduled and actual departure time. Early departures set to 0.
DepDel15	Departure Delay Indicator, 15 Minutes or More (1=Yes)
DepartureDelayGroups	Departure Delay intervals, every (15 minutes from <-15 to >180)
DepTimeBlk	CRS Departure Time Block, Hourly Intervals
TaxiOut	Taxi Out Time, in Minutes
WheelsOff	Wheels Off Time (local time: hhmm)
WheelsOn	Wheels On Time (local time: hhmm)
TaxiIn	Taxi In Time, in Minutes
CRSArrTime	CRS Arrival Time (local time: hhmm)
ArrTime	Actual Arrival Time (local time: hhmm)
ArrDelay	Difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers.
ArrDelayMinutes	Difference in minutes between scheduled and actual arrival time. Early arrivals set to 0.
ArrDel15	Arrival Delay Indicator, 15 Minutes or More (1=Yes)
ArrivalDelayGroups	Arrival Delay intervals, every (15-minutes from <-15 to >180)
ArrTimeBlk	CRS Arrival Time Block, Hourly Intervals
Cancelled	Cancelled Flight Indicator (1=Yes)
CancellationCode	Specifies The Reason For Cancellation
Diverted	Diverted Flight Indicator (1=Yes)
CRSElapsedTime	CRS Elapsed Time of Flight, in Minutes
ActualElapsedTime	Elapsed Time of Flight, in Minutes
AirTime	Flight Time, in Minutes
Flights	Number of Flights
Distance	Distance between airports (miles)
DistanceGroup	Distance Intervals, every 250 Miles, for Flight Segment
CarrierDelay	Carrier Delay, in Minutes
WeatherDelay	Weather Delay, in Minutes
NASDelay	National Air System Delay, in Minutes
SecurityDelay	Security Delay, in Minutes
LateAircraftDelay	Late Aircraft Delay, in Minutes
FirstDepTime	First Gate Departure Time at Origin Airport
TotalAddGTime	Total Ground Time Away from Gate for Gate Return or Cancelled Flight

Field Name	Description
LongestAddGTime	Longest Time Away from Gate for Gate Return or Cancelled Flight
DivAirportLandings	Number of Diverted Airport Landings
DivReachedDest	Diverted Flight Reaching Scheduled Destination Indicator (1=Yes)
DivActualElapsedTime	Elapsed Time of Diverted Flight Reaching Scheduled Destination, in Minutes. The ActualElapsedTime column remains NULL for all diverted flights.
DivArrDelay	Difference in minutes between scheduled and actual arrival time for a diverted flight reaching scheduled destination. The ArrDelay column remains NULL for all diverted flights.
DivDistance	Distance between scheduled destination and final diverted airport (miles). Value will be 0 for diverted flight reaching scheduled destination.
Div1Airport	Diverted Airport Code1
Div1AirportID	Airport ID of Diverted Airport 1. Airport ID is a Unique Key for an Airport
Div1AirportSeqID	Airport Sequence ID of Diverted Airport 1. Unique Key for Time Specific Information for an Airport
Div1WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code1
Div1TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code1
Div1LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code1
Div1WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code1
Div1TailNum	Aircraft Tail Number for Diverted Airport Code1
Div2Airport	Diverted Airport Code2
Div2AirportID	Airport ID of Diverted Airport 2. Airport ID is a Unique Key for an Airport
Div2AirportSeqID	Airport Sequence ID of Diverted Airport 2. Unique Key for Time Specific Information for an Airport
Div2WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code2
Div2TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code2
Div2LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code2
Div2WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code2
Div2TailNum	Aircraft Tail Number for Diverted Airport Code2
Div3Airport	Diverted Airport Code3
Div3AirportID	Airport ID of Diverted Airport 3. Airport ID is a Unique Key for an Airport
Div3AirportSeqID	Airport Sequence ID of Diverted Airport 3. Unique Key for Time Specific Information for an Airport
Div3WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code3
Div3TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code3
Div3LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code3
Div3WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code3
Div3TailNum	Aircraft Tail Number for Diverted Airport Code3
Div4Airport	Diverted Airport Code4
Div4AirportID	Airport ID of Diverted Airport 4. Airport ID is a Unique Key for an Airport
Div4AirportSeqID	Airport Sequence ID of Diverted Airport 4. Unique Key for Time Specific Information for an Airport
Div4WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code4
Div4TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code4
Div4LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code4

Field Name	Description
Div4WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code4
Div4TailNum	Aircraft Tail Number for Diverted Airport Code4
Div5Airport	Diverted Airport Code5
Div5AirportID	Airport ID of Diverted Airport 5. Airport ID is a Unique Key for an Airport
Div5AirportSeqID	Airport Sequence ID of Diverted Airport 5. Unique Key for Time Specific Information for an Airport
Div5WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code5
Div5TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code5
Div5LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code5
Div5WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code5
Div5TailNum	Aircraft Tail Number for Diverted Airport Code5

Appendix D: All Available Features from Processed FAA Aircraft Registration Data

Field Name	Description
N-NUMBER	Identification number assigned to aircraft.
SERIAL NUMBER	The complete aircraft serial number assigned to the aircraft by the manufacturer.
MFR MDL CODE	A code assigned to the aircraft manufacturer, model and series
ENG MFR MDL	A code assigned to the engine manufacturer and model.
YEAR MFR	Year manufactured.
TYPE REGISTRANT	3 - Corporation, 7 - LLC
NAME	The first registrant's name which appears on the Application for Registration, AC Form 8050-1.
STREET	The street address which appears on the Application for Registration, AC Form 8050-1 or the latest street address reported.
STREET2	The 2nd street address which appears on the Application for Registration, AC, Form 8050-1, or the latest street address reported.
CITY	The city name which appears on the Application for Registration, AC Form 8050-1 or the latest address reported.
STATE	The state name which appears on the Application for Registration, AC Form 8050-1 or the latest address reported.
ZIP CODE	The state name which appears on the Application for Registration, AC Form 8050-1 or the latest address reported.
REGION	Registrant's Region 1 - Eastern 2 - Southwestern 3 - Central 4 - Western-Pacific 5 - Alaskan 7 - Southern 8 - European C- Great Lakes E - New England S - Northwest Mountain
COUNTY	A code representing the county which appears on the Application for Registration.
COUNTRY	A code representing the country which appears on the Application for Registration.
LAST ACTION DATE	Format YYYY/MM/DD
CERT ISSUE DATE	Format YYYY/MM/DD
CERTIFICATION	The airworthiness certificate class which is reported on the Application for Airworthiness, FAA Form 8130-6.
TYPE AIRCRAFT	4 - Fixed wing single engine 5 - Fixed wing multi engine
TYPE ENGINE	4 - Turbo-jet 5 - Turbo-fan
STATUS CODE	V - Valid Registration
MODE S CODE	Aircraft Transponder Code
FRACT OWNER	Y - Registration has fractional ownership blank - Registration is not fractional owned
AIR WORTH DATE	Date of Airworthiness
OTHER NAMES(1)	1 st co-owner or partnership name
OTHER NAMES(2)	2 nd co-owner or partnership name
OTHER NAMES(3)	3 rd co-owner or partnership name
OTHER NAMES(4)	4 th co-owner or partnership name
OTHER NAMES(5)	5 TH co-owner or partnership name
EXPIRATION DATE	Format YYYY/MM/DD
UNIQUE ID	Unique Identification Number
KIT MFR	Kit Manufacturer Name
KIT MODEL	Kit Model Name
MODE S CODE HEX	Mode S Code in Hexidecimal Format
CODE	A code assigned to the engine manufacturer and model.

Field Name	Description
MFR	The name of the engine manufacturer.
MODEL	The name of the engine model
TYPE	Type Engine: 4 - Turbo-jet 5 - Turbo-fan
HORSEPOWER	Horsepower for engines types 1, 2, 3, 7, 8
THRUST	Thrust for engine types 4, 5, 6
CODE_aircraft	A code assigned to the aircraft manufacturer, model and series.
MFR_aircraft	Name of the aircraft manufacturer.
MODEL_aircraft	Name of the aircraft model and series.
TYPE-ACFT	Type Aircraft: 4 - Fixed wing single engine 5 - Fixed wing multi engine
TYPE-ENG	Type Engine: 4 - Turbo-jet 5 - Turbo-fan
AC-CAT	Aircraft Category Code
BUILD-CERT-IND	Builder Certification Code
NO-ENG	Number of engines on the aircraft
NO-SEATS	Maximum number of seats in the aircraft.
AC-WEIGHT	Aircraft maximum gross take off weight in pounds.
SPEED	Aircraft average cruising speed in miles per hour. This data element is not present on all records.
TC-DATA-SHEET	Type Certificate Data Sheet
TC-DATA-HOLDER	Type Certificate Data Holder

Appendix E: Causes of Delays

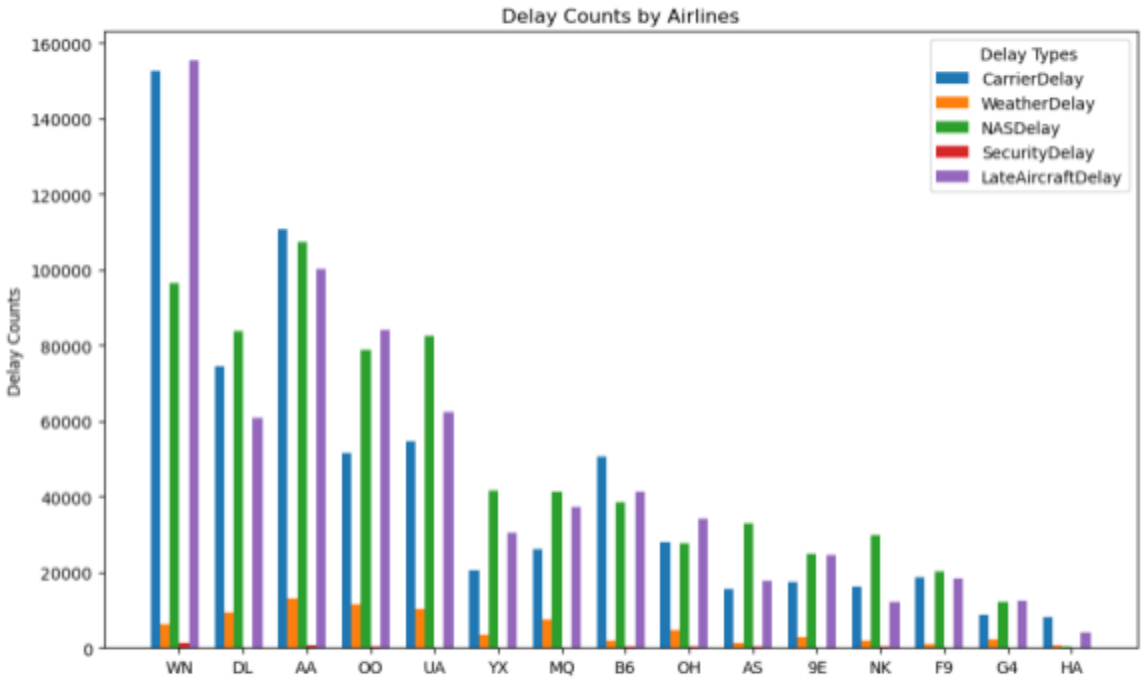


Figure E-1: Delay Counts by Airlines

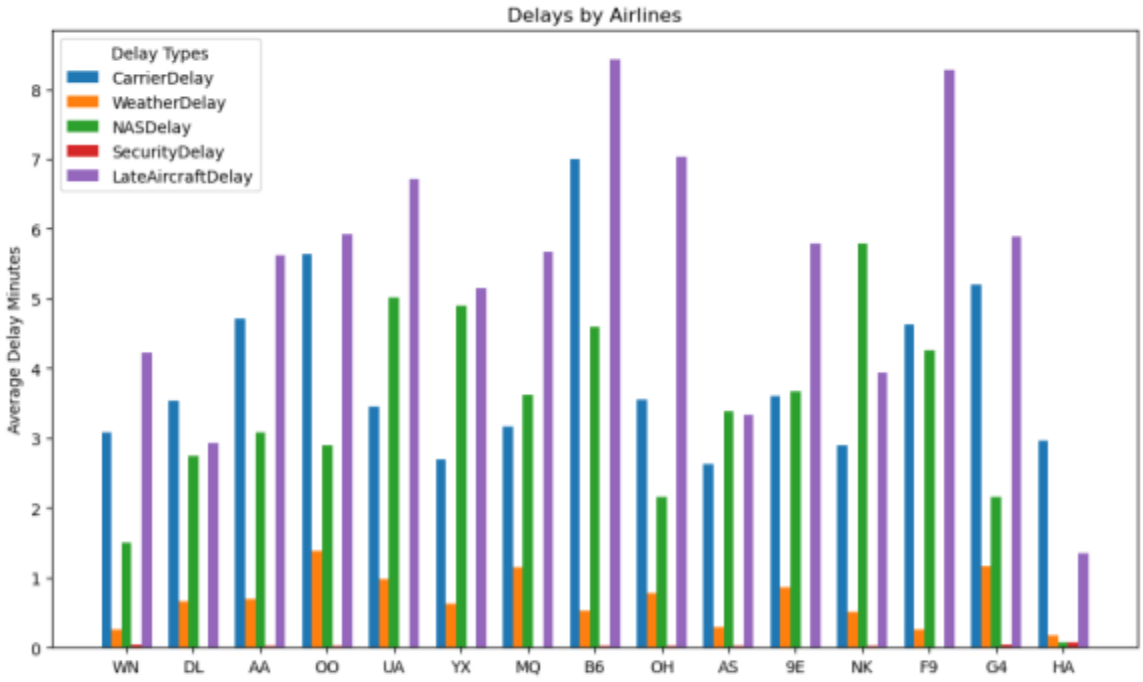


Figure E-2: Minutes of Delay by Airline and by Delay Type

Appendix F: Feature Importance

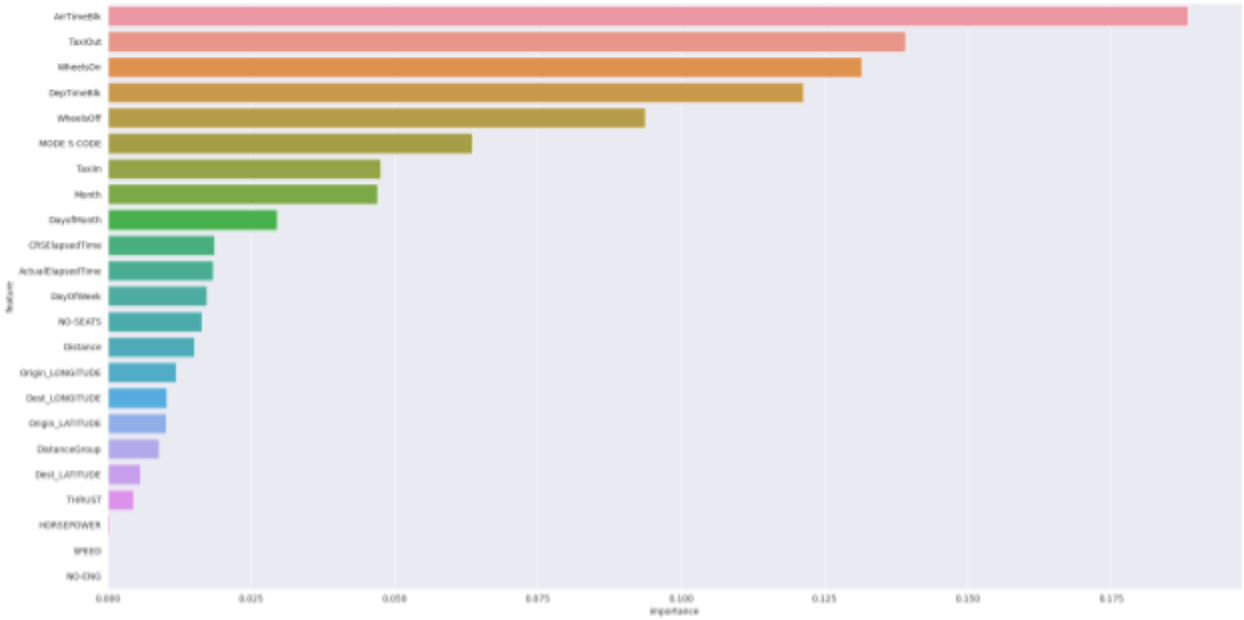


Figure F-1: XGBoost Importance Method

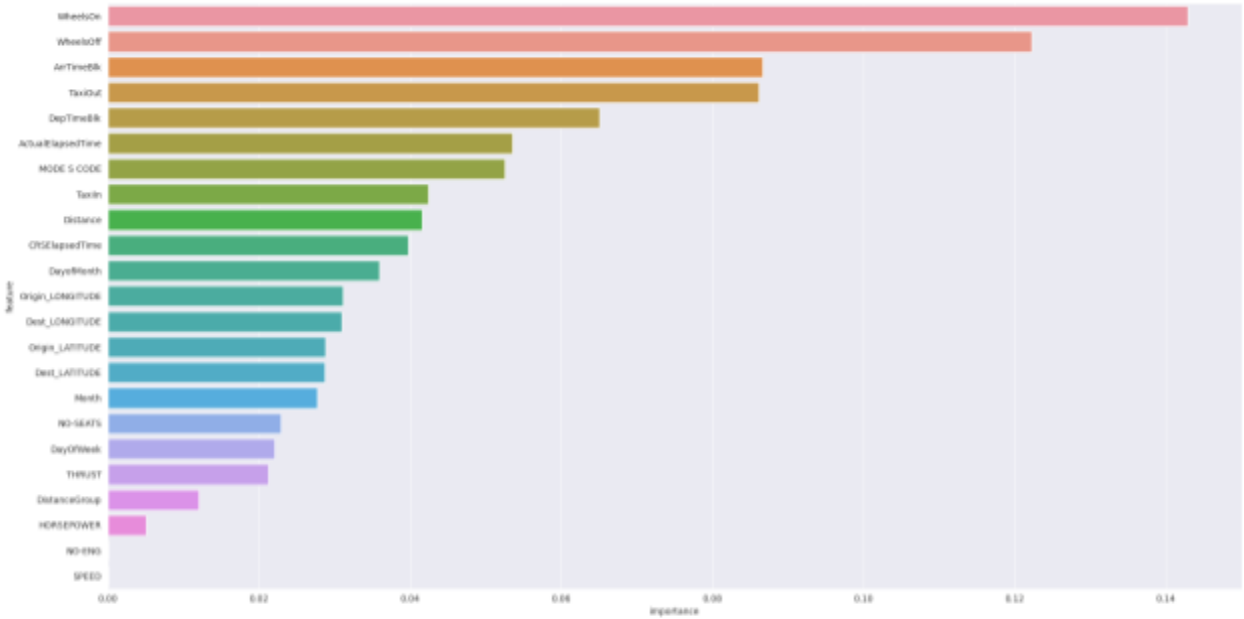
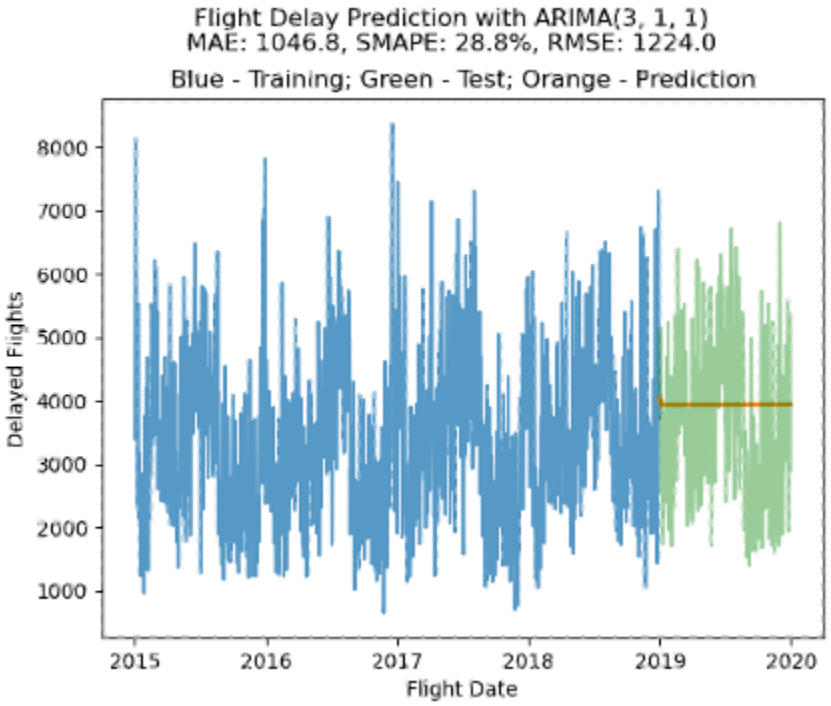
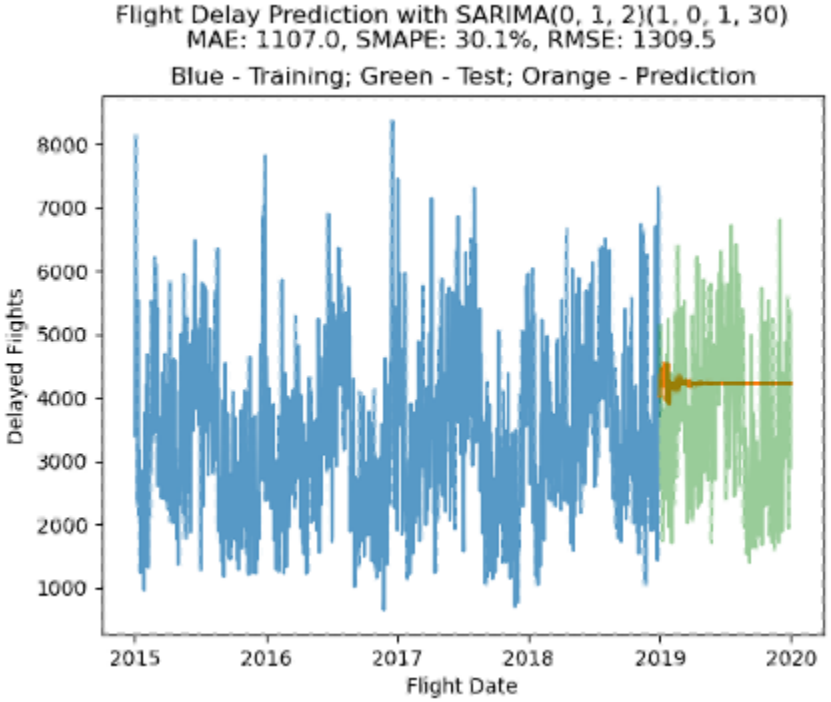


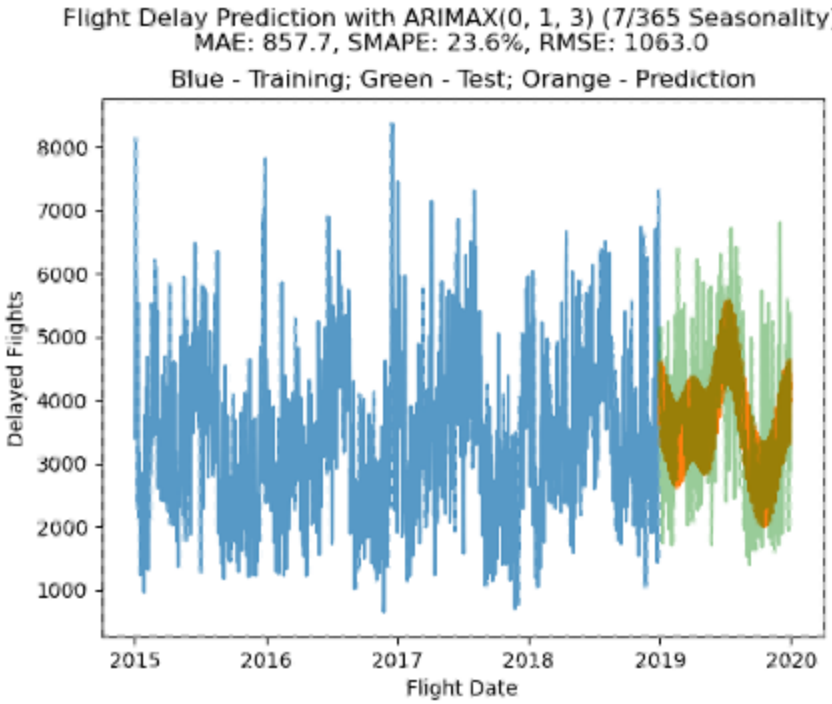
Figure F-2: Random Forest Importance Method

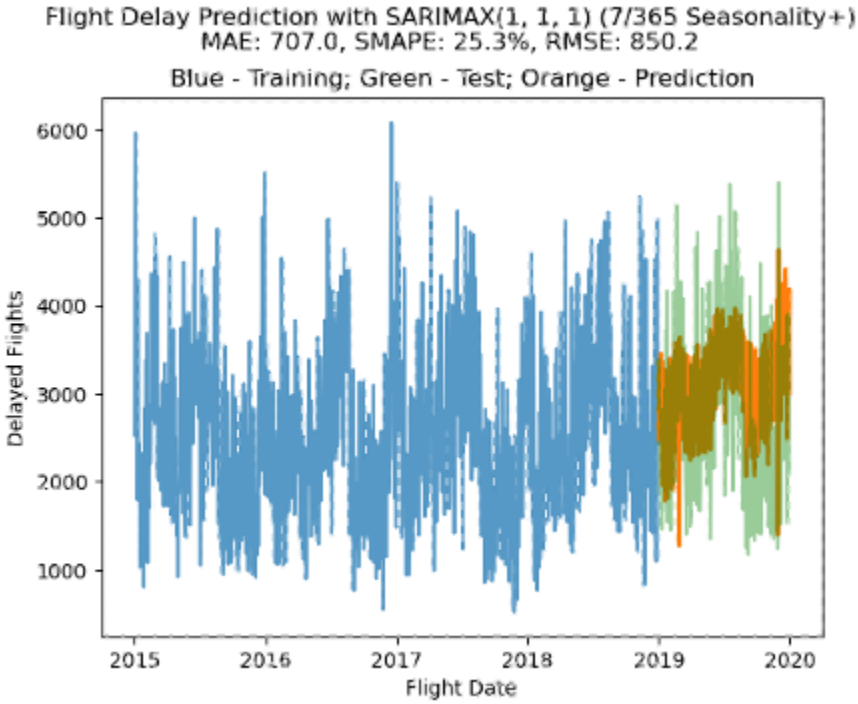
Appendix G: Time Series Visualizations

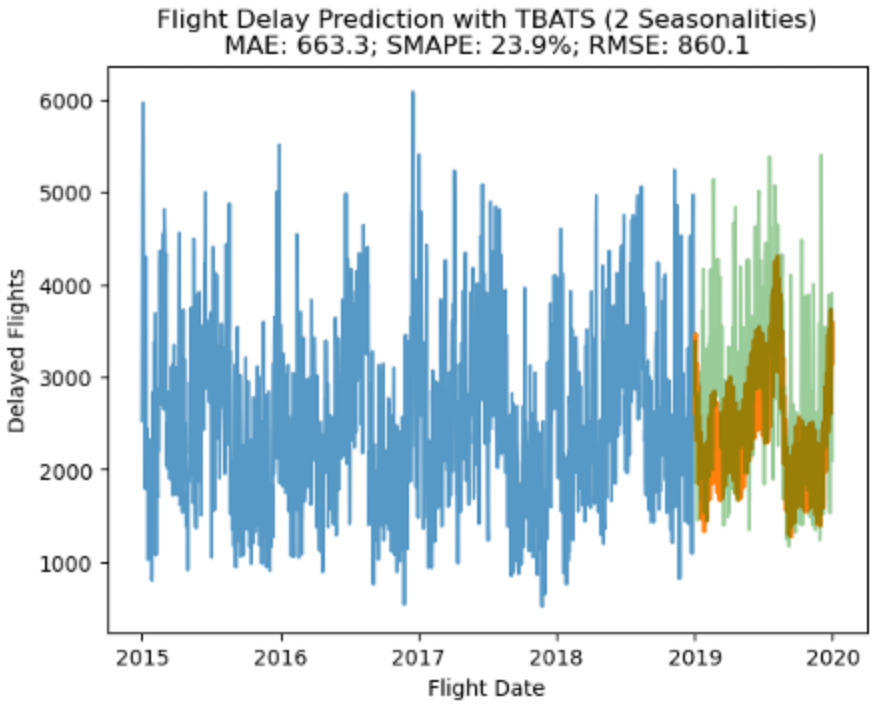
Below are larger versions of the individual visualizations shown in Figure 7 in the main body of the report.

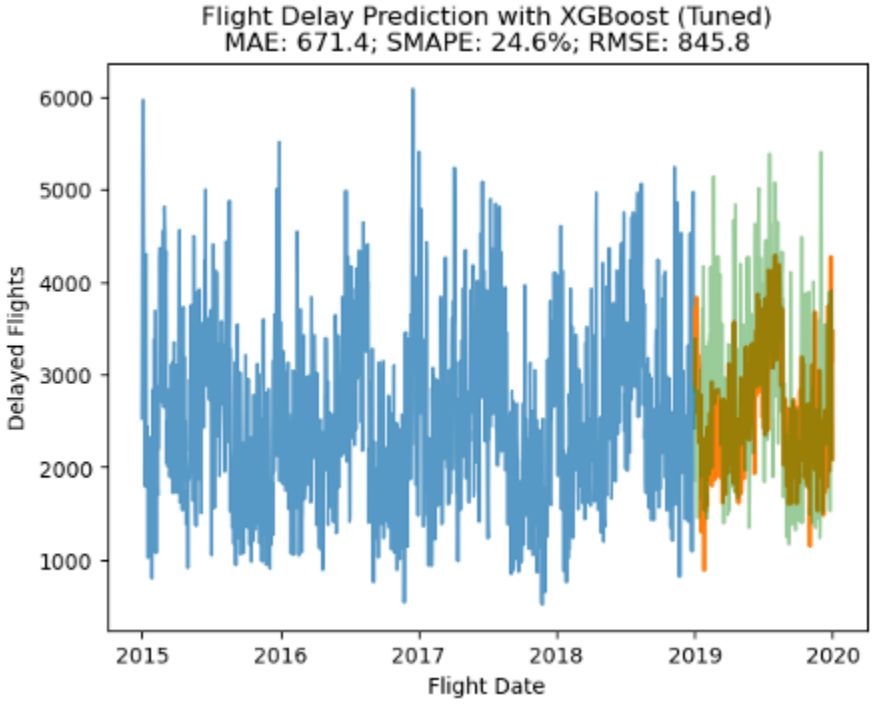





Flight Delay Prediction with ARIMAX(0, 1, 3) (7/365 Seasonality)
MAE: 857.7, SMAPE: 23.6%, RMSE: 1063.0
Blue - Training; Green - Test; Orange - Prediction
Delayed Flights
8000
7000
6000
5000
4000
3000
2000
1000
2015
2016
2017
2018
2019
2020
Flight Date


Flight Delay Prediction with SARIMAX(1, 1, 1) (7/365 Seasonality+)
MAE: 707.0, SMAPE: 25.3%, RMSE: 850.2
Blue - Training; Green - Test; Orange - Prediction
Delayed Flights
6000
5000
4000
3000
2000
1000
2015
2016
2017
2018
2019
2020
Flight Date


Flight Delay Prediction with TBATS (2 Seasonalities)
MAE: 663.3; SMAPE: 23.9%; RMSE: 860.1
Delayed Flights
6000
5000
4000
3000
2000
1000
2015
2016
2017
2018
2019
2020
Flight Date


Flight Delay Prediction with XGBoost (Tuned)
MAE: 671.4; SMAPE: 24.6%; RMSE: 845.8
Delayed Flights
6000
5000
4000
3000
2000
1000
2015
2016
2017
2018
2019
2020
Flight Date